(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
17 May 2001 (17.05.2001)

PCT

(10) International Publication Number
WO 01/35211 A2

(51) International Patent Classification[7]: G06F 9/00

(21) International Application Number: PCT/US00/30693

(22) International Filing Date: 8 November 2000 (08.11.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/436,885 9 November 1999 (09.11.1999) US

(71) Applicant: JARNA, INC. [US/US]; 1800 El Camino Real, Suite D, 2nd floor, Menlo Park, CA 94027 (US).

(72) Inventor: KAPOOR, Sanjay; 475 Cumulus Avenue, #32, Sunnyvale, CA 94087 (US).

(74) Agents: VAUGHAN, Daniel et al.; Park & Vaughan LLP, 702 Marshall Street, Suite 310, Redwood City, CA 94063 (US).

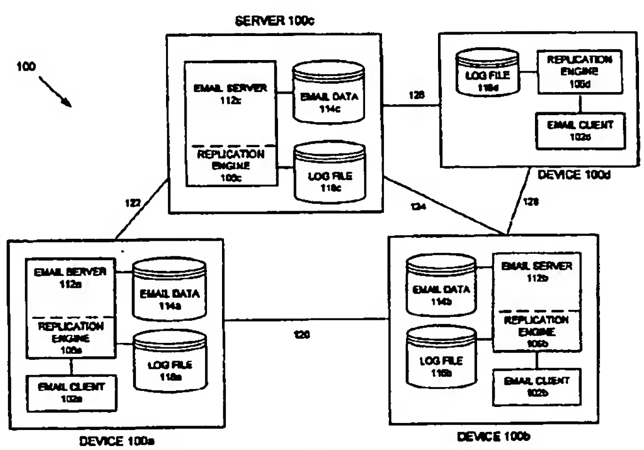
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *Without international search report and to be republished upon receipt of that report.*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: SYNCHRONIZING DATA AMONG MULTIPLE DEVICES IN A PEER-TO-PEER ENVIRONMENT



(57) Abstract: A system for synchronizing data in a peer-to-peer environment includes multiple computing devices sharing a set of data. Each device includes a replication module for logging and trading data changes with other devices. A device having sufficient resources may also include a local version of the shared data and an application server that allows access to the data even while the device is disconnected from other devices. The replication module logs a description of each operation that alters the data. When one device is connected to a second device in the system to synchronize data, the devices' replication modules exchange log entries. For example, one device receives from a second device data changes made by the second device and other devices that already synchronized with the second device. Thus, the method of data synchronization is transitive and allows a device to fully update its local copy of the shared data by connecting to just one other device, which may or may not be a central server.

WO 01/35211 A2

SYNCHRONIZING DATA AMONG MULTIPLE DEVICES IN A PEER-TO-PEER ENVIRONMENT

BACKGROUND

This invention relates to the field of computer systems. More particularly, a system and methods are provided for synchronizing data shared among multiple computing devices on a peer-to-peer basis.

The need to share data among multiple computing devices is a common requirement. One person may employ several different devices (e.g., desktop computer, laptop, hand-held) for different purposes or at different locations, but yet require some data be available on all of them. Similarly, several people may employ different devices for common purposes (e.g., in a work group) and wish or need to have access to the same data at each device.

Effective synchronization or management of the shared data is a common goal in both types of environments. In particular, it is desirable to have the same, or nearly the same, data available to applications common to multiple devices. Thus, a single user may wish to access his or her email or personal calendar, with up-to-date data, regardless of which device is most convenient to use. Similarly, in a file sharing arrangement among a group of workers, it is preferable to maintain nearly identical views of the file system for each worker. Further, the applications among which data are shared may or may not be identical. Thus, multiple users may wish to share an address book or set of electronic mail data even though they each use different electronic mail applications or different operating systems.

As the number of devices and applications (e.g., electronic mail, calendar, address book, web browsing, file access) used on the devices grows, so too does the difficulty of managing the data. In a fully networked environment in which each of multiple devices is always, or nearly always, connected to other devices or a central computer system (e.g., a server), it may be difficult to ensure that each user or device is using or modifying the latest version of shared data. For example, present data management techniques may require each user or local device to ensure that it downloads or copies the latest version or receives all changes made to shared data since a last access. An automated system for

ensuring data synchronization could greatly reduce the possibility of having different users create different versions of the data.

In addition, in an environment in which one or more computing devices are rarely, infrequently or irregularly connected to other devices participating in a data sharing arrangement, data management becomes even more difficult. Presently, an effective solution is lacking for the need to synchronize data among multiple devices in such an environment. In particular, in this type of environment one may wish to operate the disconnected device with virtually the same functionality as when connected. And, when one data-sharing device does connect to another device, the device should be able to pass on changes that were made to its data while disconnected and receive data changes made by other devices as well. Present solutions do not provide for this capability.

Existing methods of synchronizing data generally rely upon a centralized management scheme. In particular, a central computer system (e.g., a server in a client/server architecture) is always connected to or reachable from a number of networked devices and stores a set of data useful to the devices. Each device downloads and manipulates the data as necessary. If changed, the data must be uploaded to the central server before it can be shared with the other devices. In one such solution, data must be "published" to a web server before it is available to other devices, which must access the web server in order to obtain the data.

And, even with a centralized data synchronization scheme, it may be difficult if not impossible to share data among incompatible applications or platforms. For example, users of Microsoft Outlook cannot easily share electronic mail with users of Netscape Messenger, particularly if one is operating in a Windows environment and the other in an Apple Macintosh environment.

Thus, in centralized solutions to the need for data synchronization a central server merely acts as a location to store the data. Every time a device changes the data, every other device must access the server in order to retrieve the updated data. The devices can synchronize only with the central server; they cannot synchronize with each other. This method is therefore only as robust as the central server. Further, a communication bottleneck is created at the central server and access to the data is limited to the interface (e.g., a web browser) employed or accepted by the server.

Another method of data synchronization is limited to two devices at a time, such that one device can provide its data changes to one other device and vice-versa. Each device must separately connect to every device that may have data changes.

Thus, present solutions to the need for data synchronization among multiple computing devices are inefficient, inflexible and unsuitable for environments in which one or more devices may be regularly disconnected. What is needed then is a system for synchronizing data among peer devices in a manner that does not require frequent communication between one device and every other device so that changes made to shared data by one device may propagate to all devices. In this system users will be able to synchronize data across different applications and operating systems.

SUMMARY

In one embodiment of the invention a system and methods are provided for synchronizing data shared among multiple devices operating a common application in a peer-to-peer environment. Data may be synchronized at regular intervals or in real-time, and may be synchronized among different applications and operating systems.

In this embodiment each device includes a processor for operating an application client module for a user and a replication engine for managing the synchronization of changes to the shared data. The replication engine records descriptions of such changes in a log file. Devices having sufficient resources (e.g., desktop computers, workstations) also store a local version of the shared data. Each device's locally stored shared data may be tailored to a user's needs or the device's resources by including only a subset of all the data. Devices with sufficient resources also include an application server for accessing and manipulating the local version of the data. Thus, in this embodiment of the invention a device may be configured to operate an application with complete, or nearly complete, functionality even when disconnected from all other devices. Low-resource devices (e.g., hand-held or palm-top computers) may rely on access to another device's application server and/or local copy of the shared data.

In another embodiment of the invention a method is provided for synchronizing data among devices in a system such as that described above. In this embodiment any two or more devices may, on a peer-to-peer basis, synchronize their locally stored data. A server having high availability may be provided, however, so that a device will virtually

always be able to send data updates to or receive data updates from at least one other device.

As a device operates an application having data shared with other devices, the device's replication engine records descriptions of changes to the data made by the device's application server. In one embodiment the application server includes or is tightly coupled with the replication engine; in another embodiment the replication engine monitors the application server's actions. The descriptions are recorded as entries in a log file, and include information such as a timestamp identifying the time the description was recorded, an identifier of the device that performed the change, the application that caused the change (e.g., electronic mail, calendar, address book, file system) and the type of change (e.g., update, delete, new). Each entry may also include one or more fields of application-specific information that further describes the change and may help determine whether one change conflicts with another and, if so, how to resolve the conflict. Further, in one embodiment of the invention a log file entry may include the data that changed (e.g., a new or altered calendar entry).

When one device connects to another for data synchronization purposes, each device determines the most recent log file entry it has that it can be sure has been provided to the other node. Each then sends to each other log entries that the other may not have. A device then reviews each received entry, discards ones that it already has, and determines whether any of the data changes described in the newly received entries conflict with other data changes. If so, the conflicts are resolved, either automatically or with user intervention. Accepted entries are then recorded in each device's own log file. Depending on the configuration of each device, it may then apply to its local copy of the shared data any subset of the data changes described in the entries that it recorded.

In this embodiment, a device sends to another device log file entries corresponding not only to data changes that it performed, but also data changes performed by other devices with which it has synchronized. Thus, this method of data synchronization is transitive and data changes are propagated throughout the system without having to use a central server.

In one alternative embodiment of the invention, a device may synchronize with multiple other devices simultaneously.

DESCRIPTION OF THE FIGURES

FIG. 1 is a block diagram depicting a system for synchronizing data among peer devices in accordance with an embodiment of the present invention.

FIG. 2 is a flowchart illustrating one method of operating a device for synchronizing data in accordance with an embodiment of the invention.

FIG. 3 depicts one form of a time table for determining which log file entries one device must send to another, in accordance with an embodiment of the present invention.

FIG. 4 is a flowchart demonstrating one method of updating a device's local version of shared data in accordance with an embodiment of the present invention.

DETAILED DESCRIPTION

The following description is presented to enable any person skilled in the art to make and use the invention, and is provided in the context of particular applications of the invention and their requirements. Various modifications to the disclosed embodiments will be readily apparent to those skilled in the art and the general principles defined herein may be applied to other embodiments and applications without departing from the spirit and scope of the present invention. Thus, the present invention is not intended to be limited to the embodiments shown, but is to be accorded the widest scope consistent with the principles and features disclosed herein.

The program environment in which a present embodiment of the invention is executed illustratively incorporates a general-purpose computer or a special purpose device such as a hand-held computer. Details of such devices (e.g., processor, memory, data storage and display) are well known and are omitted for the sake of clarity.

It should also be understood that the techniques of the present invention might be implemented using a variety of technologies. For example, the methods described herein may be implemented in software executing on a computer system, or implemented in hardware utilizing either a combination of microprocessors or other specially designed application specific integrated circuits, programmable logic devices, or various combinations thereof. In particular, the methods described herein may be implemented by a series of computer-executable instructions residing on a storage medium such as a carrier wave, disk drive, or computer-readable medium. Exemplary forms of carrier waves may take the form of electrical, electromagnetic or optical signals conveying digital data streams along a local network or a publicly accessible network such as the Internet.

Introduction

In one embodiment of the invention a system and method are provided for synchronizing data that is shared among multiple computing devices. The devices may include various types of computers (e.g., desktop, laptop, hand-held) and other mechanisms possessing a processor, data storage and means for connecting to at least one other such device. The devices among which data is synchronized may be used by a single person (i.e., at different locations or times) or may be used by several people performing related or complementary tasks.

By synchronizing data among participating devices, the data may be accessed and manipulated by any individual participating device. Virtually any type of data used by programs common to two or more of the multiple devices may be synchronized. Some of the types of data that may be shared in an embodiment of the invention include electronic mail, calendar data, an address book, all or part of a file system, graphics, bookmarks for a web browser, etc.

Advantageously, the connection patterns of devices synchronizing shared data need not be regular and need not even be known in order to implement an embodiment of the invention. Of particular note, the devices need not be continuously or even frequently connected to each other or to a central system (e.g., a central server). In addition, in one embodiment of the invention the devices and application programs that use the synchronized data are operable (e.g., the data can be viewed and/or altered) even while the device is disconnected from other devices. Illustratively, a small protocol or application server (e.g., approximately 100-300 kilobytes) is installed on each device in order to make an application program functional while disconnected. And, because of the peer-to-peer nature of the devices in this embodiment, a device need only connect to one other device – and not a particular one such as a central server – in order to pass on its data changes and receive updates to the data that are generated by other devices. Devices may be added and removed dynamically while the system is in use.

As described below, the devices may be configured to automatically replicate or synchronize their data soon after being connected (e.g., via a network) to a peer device. Illustratively, two synchronizing devices exchange log file entries that describe data changes that each device knows of (e.g., performed by the sending device or a device with which the sender previously synchronized). Each device then determines whether any of

the data changes it has been informed of conflict with other data changes it knows of. Conflicts are resolved and, depending on the configuration of a device, it may retrieve (e.g., a new email message, an altered address entry) or apply (e.g., a renamed file) the actual data changes. A virtual file system is used in one embodiment of the invention in order to ensure uniform naming of shared data.

In another embodiment of the invention data may be automatically synchronized among devices in a fully or always connected environment. In other words, in a network environment in which two or more devices are always or almost always inter-connected, data may be automatically synchronized (e.g., at programmable time intervals or upon specified events). Each participating device is thus relieved of the responsibility of individually ensuring that it retrieves new or changed data.

In one or more of the embodiments described below a method for synchronizing data may apply application-specific information in order to further optimize the efficiency of data synchronization.

As mentioned above, data synchronization may be performed in a transitive manner. In other words, each device participating in the synchronization of a set of data passes not only its own data changes when synchronizing with another device, but also sends data changes it learned of from other devices. Transitive data synchronization therefore enables a device's data updates to be propagated throughout the system of participating devices by connecting to just one other device.

Although embodiments of the invention discussed below primarily describe the synchronization of data between two devices, virtually any number of devices may participate in a single synchronization event. For example, a single device may transmit its data changes to multiple other devices either in multiple unicast operations or one multicast operation.

A System for Synchronizing Data Among Multiple Computing Devices

FIG. 1 is a block diagram depicting one system for synchronizing data among multiple devices according to one embodiment of the invention. In FIG. 1, system 100 includes computing devices 100a, 100b, 100c and 100d. The illustrated embodiment of the invention is not limited to a particular number of participating devices. For example, in an embodiment of the invention in which a wide-area network such as the Internet is

used to inter-connect participating devices, virtually any number of devices may included in system 100.

Illustratively, device 100a may be a laptop computer, device 100b a desktop computer or workstation, device 100c a server (e.g., web server, network server) and device 100d a low-resource device such as a hand-held or palm-top computer. Links 120, 122, 124, 126 and 128 may be any type of links (e.g., dedicated, on-demand, wired, wireless) for connecting one device to another and may be part of a network accessible to synchronizing devices. In particular, in one environment one or more of links 120-128 form part of network, such as the Internet, that is compatible with the Internet Protocol (IP) and either the TCP (Transmission Control Protocol) or UDP (User Datagram Protocol). Links 120-128 in this embodiment thus enable one device in system 100 to exchange data with another device and may employ virtually any configuration and medium. Any or all of the links between devices in system 100 may be active on only an irregular or infrequent basis.

Each device in FIG. 1 includes sufficient resources (e.g., processor, data storage, network access) to enable it to store and process data and communicate with at least one other device in system 100. Further, all or most devices in the system operate one or more applications (e.g., electronic mail, calendar, address book) in common with at least one other device. Thus, in FIG. 1, devices 100a, 100b and 100d operate an electronic mail program represented by email clients 102a, 102b and 102d. In an alternative embodiment of the invention devices may have multiple applications in common and therefore require synchronization of multiple sets of data.

Each device in system 100 also includes an application or protocol server module with which to access a local version of shared data. For example, email servers 112a, 112b perform operations on email databases 114a, 114b at the request of email clients 102a, 102b. Client modules on low-resource devices such as device 100d may rely on application servers on other devices in system 100, such as device 100b or server 100c. A separate server module may be installed on a device for each shared application, or one server module may service multiple applications.

Each application server is relatively small compared to servers previously used to service client requests. In particular, application servers installed on synchronizing devices in a present embodiment of the invention are configured primarily to provide essential data services that allow a device to access and/or manipulate data even while

disconnected from other devices. Also, however, an application server on one device allows another device to request data changes made or recorded on the one device to be retrieved and provided to the other device. One skilled in the art will appreciate the novelty and significance of installing an application server on a user's device and recognize the flexibility that such a configuration affords the user concerning operation of the device and synchronization of data shared among devices.

Replication engines 106a, 106b, 106c and 106d manage the replication and synchronization of data in system 100. For example, when device 100a connects to another device in the system after a period of disconnected operation, replication engine 106a ensures that changes to the shared data are transmitted to the connected device. Thus, email database 114, represented in FIG. 1 by its local versions 114a, 114b and 114c comprises data to be synchronized among the devices of system 100.

Illustratively, a replication engine module is composed of a series of instructions executable by a device's processor to transmit and/or receive data and/or data changes. In FIG. 1, replication engines 106a, 106b and 106c are depicted as part of email servers 112a, 112b and 112c. However, one skilled in the art will appreciate that in an alternative embodiment of the invention a replication engine may be distinct from or coupled with application client modules (e.g., email client 102a) and/or application server modules (e.g., email server 112a). For example, a low-resource device such as device 100d operates replication engine 106d but has no separate application server for its shared application(s). Instead, it accesses shared data through email server 112a, 112b or 112c.

Finally, each device includes a log file for recording local application operations or transactions that change the shared data. More specifically, each time the shared data is altered (e.g., a new message is received, a message is deleted or moved) an entry is recorded in the log file of the device that performed or initiated the alteration. During data synchronization with another device, a method of which is described below, the log file entry is provided to the other device. The receiving device then determines whether the operation conflicts with other known data alterations and resolves any such conflict. If the operation is to be retained, the receiving device records the operation in its own log file and may, depending on its configuration, apply the change to its local data. In system 100, replication engines 106a, 106b, 106c and 106d manage log files 116a, 116b, 116c and 116d.

Illustratively, application servers and replications engines installed on the same device are tightly coupled. Thus, as the application server services requests from an application client it notifies the replication engine of transactions that alter shared data so that the replication engine may record the entry. Advantageously, each log file entry contains sufficient information to fully identify and chronologically order data changes. Each entry also identifies the location and nature of the associated data change so that other devices may apply the change and keep the shared data in synchronization.

In the illustrated embodiment of the invention device 100c is a computer system or server that is virtually always online and accessible to devices 100a, 100b and 100d. Server 100c thus provides the ability for any other device in system 100 to synchronize email data at virtually any time. When configured in this manner, server 100c may not include client modules for each application for which data is being shared. Alternatively, however, server 100c may be configured to accept connections from "thin" devices (e.g., web browsers, hand-held computers, smart or web-enabled telephones) and other devices that lack the resources to store full local versions of shared data, such as device 100d.

The configuration of server 100c thus allows thin devices to participate in a data synchronization system. The components of server 100c (e.g., application client modules, application servers, data storage, etc.) may therefore be used to different degrees by different devices. For example, email data 114c on server 100c may serve as the local version of shared data for device 100d or any other device that is not configured to store any or all of the shared data. Devices having sufficient resources, however, maintain local versions of the email data.

Thus, system 100 of FIG. 1 facilitates not only the distributed sharing of data (i.e., among multiple devices) but also the partial sharing of data. In other words, any device in system 100 may be configured, depending on its resources, to store any subset of data shared among common applications. A device with sufficient resources, such as a workstation or desktop computer may be configured to store all shared data locally. Another device may be configured to share (and synchronize) data for fewer than all shared applications – e.g., electronic mail but not an address book. Yet another device may possess limited storage space and therefore be configured to store only files of a certain size or type or within a given range of sizes and types. As already described, a device (e.g., device 100d) may avoid storing any local data and, instead, use another device's local version of the stored data when necessary.

Devices participating in a synchronization scheme in an embodiment of the invention may operate a variety of operating systems and user interfaces. For example, one device may operate Windows 98, another may use Windows NT, another may run a version of Unix, yet another may operate Linux, etc. Advantageously, the replication engine and application servers installed on each device are configured to work with whatever user interface or operating system the device normally uses. The devices may interconnect via different network operating systems and protocols as well.

One skilled in the art will perceive clear differences between the environment of FIG. 1 and a typical client/server environment. For example, in a typical client/server architecture client systems do not include application servers with which to access local copies of shared data. In fact, application servers for application programs in a client/server environment are too complex and require too much management to install on each user device. Instead, an email database resides on a central system having an email server, to which each client device connects with email requests. Therefore, in a client/server environment every request from a client, and the response to the request, must transit the distance between the client system and the server even if the request does not change any data. And, the data returned in response to a client request must be sent in full; the server cannot just send or describe a change that occurred in the data.

Also, in typical client/server architectures the server cannot differentiate between clients, whether operated by the same or different users. Thus, each client must separately ensure that it receives the latest version of shared data or all changes that have occurred since a last update. For example, a client may access or retrieve the status or states of a full set of shared data. The client may then compare the current (i.e., received) status of each data item (e.g., different electronic mail messages) with the last known status of the item. For data items that have changed, the client may then download the necessary changes.

A system such as system 100 may be constructed by individually configuring and adding devices, or the devices and system 100 may be configured all at once. Illustratively, server 100c may be first configured with replication engine 106c in order to provide a default synchronization partner as each device is added to the system.

In one embodiment of the invention a device is automatically added to a system for synchronizing data, such as system 100, when the device first connects to server 100c or another participating device. Illustratively, each device within the system, or each

device within a particular group of devices sharing a set of data, is given a unique name before or during this first connection. Server 100c then recognizes the device as a new member of the system or a group within the system, ensures that the appropriate application servers (if any) and replication engine(s) are installed and provides a first copy of shared data.

Alternatively, however, a device may be partially or fully configured offline (i.e., disconnected from any other device) and then connected in order to register the device, complete the configuration, and/or being data synchronization.

The local versions of shared data may or may not be complete copies of the data. As already described, a low-resource device (e.g., a hand-held computer) may be configured so as not to store large files but rather access them remotely if the need arises. Another device may be configured to avoid copying specific types of files (e.g., graphics, animation) unless a user of the device specifically requests access to the file. Further, because a participating device may be operated in a disconnected mode, during such operation the device is not notified of data changes and cannot pass on its own changes. Thus, the degree to which the local versions of data match each other depends not only on the configuration of each device (i.e., what data it is configured to store locally) but how frequently it is connected to another device. Illustratively, however, the manner in which one device connects to another (e.g., direct connect, through multiple segments of a network) and the media used to make the connection do not limit the synchronization of data.

In one embodiment of the invention the replication or synchronization of shared data among many devices is facilitated by informing all devices of every data change (e.g., new file, modified calendar entry, renamed email message) made by a device. In particular, in the embodiment of the invention illustrated in FIG. 1, data changes are learned through the exchange of log file entries. Illustratively, each entry in a log file (e.g., log file 116a of FIG. 1) adheres to a standard format described below. During data synchronization one device passes some or all of its log file entries to another device in order to inform that device of data changes that the device may not have already learned. The device receiving the log entries may then request the new or modified data and apply all or any subset of the changes.

A virtual file system is maintained within system 100 in order to uniquely identify every file that is shared among multiple devices. As described shortly, log file entries

identify files, directories, messages, and other entities by their virtual names rather than their local names, which may differ on each device. A virtual file name may have the form //domain/shared_directory/file_name in one embodiment of the invention.

Each virtual file is physically stored on at least one of the devices of system 100, with a local file name that corresponds to the naming convention of the device (e.g., Windows, Unix, Linux). Every operation that modifies a file (e.g., create, update, move) is logged in the modifying device's log file and then passed to other devices as described in a following section. Each entry in this embodiment identifies the file and the device that modified it. Thus, every device can locate a virtual file system entity (i.e., if not stored locally) by searching its log file for the virtual file name and examining pertinent entries to determine which device(s) most recently modified and/or stored the file.

In one embodiment of the invention log file entries include the following fields. In alternative embodiments of the invention any subset of this information may be included in a log file entry, as may any other information that would facilitate data synchronization.

Field	Description
Timestamp	Time at which the transaction was performed or entered in the log file.
Node	Identity of the node on which the transaction occurred.
Application	Application in which the transaction occurred (e.g., email, calendar)
Operation	Type of transaction (e.g., new, update, rename, delete)
Metadata	Application and/or Operation specific information (e.g., old and new names of a renamed file)

In this embodiment, the format of the timestamp field is uniform among all participating devices and includes information indicating the year, month, day, hour, minute and second of each transaction (e.g., 19990529161504). Timestamps may be even more accurate in another embodiment by measuring time to the tenth, hundredth or some other fraction of a second. Illustratively, timestamps are normalized to GMT (Greenwich Mean Time) or some other common time reference.

The node field of an entry in this embodiment identifies the device that performed the recorded operation. Each device is thus assigned a unique name within a group of

synchronizing devices to which it subscribes. A node field entry may include just the device's name or may also include a domain (e.g., group) name or other indicia.

In one embodiment of the invention the application field of a log file entry may have any of the following values, which correspond to applications operating and sharing data on one or more synchronizing devices: file, email, calendar, address or bookmark. File refers to file system and indicates that some change was made to a file or directory. The applications that are synchronized in a particular embodiment of the invention are not limited to the foregoing. For example, word processing may be implemented as a separate application or may be considered part of the file application.

Various operations are defined to identify the types of data changes that may occur within a particular application. For example, in one embodiment of the invention operations for file applications may include new, move (or rename), update (or modify) and delete, and may correspond to a specific file or directory or a pattern of files or directories (e.g., using wildcard characters). Email operations may include the same options as the file application but may be applied to either a message or a folder. Calendar, address book and bookmark operations may include new, modify and delete. One skilled in the art will appreciate that the various types of transactions that may be conducted in a particular application may be described using many different labels. Thus, the possible values for an operation field in a particular embodiment of the invention are by no means limited to those enumerated here.

Finally, one or more fields of a log file entry may include application and/or operation specific information to help identify or apply a particular transaction. For example, when a file, message, or address is deleted, the name of the file, message or address is provided in one such field. When a file is modified (e.g., by a word processing program), the name (e.g., the virtual name) of the file is recorded and, possibly, a timestamp identifying the time of the change or a value reporting the new size of the file. In sum, whatever application-specific or operation-specific information may be helpful to another device to identify, understand or apply a data change may be included in these fields.

The format of a log file entry and acceptable values for the fields of a log file entry may be different in an alternative embodiment of the invention. For example, a new type of application may be shared among devices, thus creating a new value for the application field.

<u>**Operating a System for Synchronizing Data Among Multiple Devices**</u>

In one embodiment of the invention a method is provided for operating a system to synchronize data among multiple computing devices in a peer-to-peer environment. In particular, multiple devices are configured to store local versions of data used by one or more applications common to all or a subset of the participating devices. Advantageously, this method of operation allows the devices to operate a common or shared application while disconnected from any other devices and easily update its version of the data by connecting to and synchronizing with any other device. This method is therefore suitable for use in an environment in which one or more devices operate regularly or primarily in a disconnected mode.

In this embodiment, one device informs another of data changes it knows of by providing corresponding entries from a log file (e.g., log file 116a of FIG. 1). For example, when device 100a connects to device 100b (e.g., after operating in a disconnected mode), each device transmits to the other the entries in its log file that it determines the other device may not have. The update procedure is transitive in that device 100a will inform device 100b of data updates it knows of that were performed on other devices in system 100 (e.g., server 100c or device 100d).

Because the log file entries are small and may consist of just metadata, they may be rapidly and efficiently exchanged. Although a log file entry just describes a data update in a present embodiment of the invention, in an alternative embodiment a log file entry may include certain kinds (e.g., within a certain size range, of a particular application or operation type) of data changes. As described in the following section, after receiving exchanging log file entries, synchronizing devices may then apply or reject data updates depending upon whether the updated conflict with other operations and depending on how the devices are configured (e.g., depending on the subset of shared data they store locally).

Thus, in the presently described method of data synchronization a device may be fully updated by connecting to any other device in the system. As described in the previous section, one or more devices (e.g., server 100c) may be configured for high availability so that a disconnected device will always have at least one other device with which it may synchronize at virtually any time.

Illustratively, the synchronization process is automatically performed when a device is connected to a participating device. Alternatively, however, devices that tend to be connected to the system more often than disconnected may be configured to synchronize with a programmable regularity measured by time, system events (e.g., every time 50 data updates are performed on a device) or some other criterion.

In one method of the invention, a replication engine is installed on each device participating in the system. The replication engine may comprise a separate series of executable instructions or may be incorporated as part of another software module installed on the device. In one embodiment of the invention a replication engine is incorporated into a local application server (e.g., an electronic mail server) installed on participating devices. In particular, an application server may be created to handle all or a subset of all data operations an application client may request and a replication engine may be incorporated into the server or be tightly coupled with it.

Each time a device user changes the application data (e.g., by marking a message as read, moving a message to another folder, deleting a message) the replication engine logs the event in a log file stored on the device or in a location accessible to the device. Illustratively, an application client module receives the user's input and submits a corresponding request to the application server installed on the device. The application server module not only satisfies the request but informs the replication engine of the transaction if it involves a change to the data. In one alternative embodiment of the invention, a replication engine may be configured to monitor data operations on its own (e.g., without being separately informed by an application server).

When the device is connected to another, it is the replication engine that handles data synchronization with the other device by first exchanging log file entries. If a data change learned from another device is to be applied locally, the replication engine may contact the application server and/or replication engine of the device on which the change is stored in order to retrieve the data change (e.g., new electronic mail message, altered file).

In one embodiment of the invention data may be encrypted or otherwise protected. For example, the data changes and/or other information transmitted from one device to another may be encrypted during transit. Or, a device's data may be stored in an encrypted format and exchanged with other devices in the same or other secure format.

Various forms of encryption and data security are suitable, as will be apparent to those of ordinary skill in the art.

FIG. 2 is a flowchart demonstrating one method of configuring and operating device 100a within system 100 according to one embodiment of the invention.

Prior to state 200 in FIG. 2, server 100c is configured for each application having data to be synchronized in system 100. This may, for example, require the installation of an application server and/or application client for each such application. In addition, a replication engine and a first set of data for the application(s) is stored on the server.

In an alternative embodiment, device 100c may be just another device in system 100, not necessarily one dedicated to making data synchronization always available. Device 100c may be configured in this embodiment to accept device connections in various forms, such as through a network connection (e.g., the Internet), via modem, etc. Device 100c may accept or employ one or more interfaces compatible with the devices of system 100c, such as a network operating system, a web browser, etc.

In state 200 of FIG. 2, device 100a (e.g., a desktop, laptop, hand-held computer) of FIG. 1 is configured according to a user's needs. In particular, one or more applications having data to be shared among the devices in system 100 are installed and configured. Thus, email client 102a is configured during state 200.

In state 202 application or protocol servers for the user's applications that are to be synchronized are installed, along with one or more replication engines. Each application server may incorporate a separate replication engine or multiple application servers on one computing device may employ a single replication engine. A separate application or protocol server module may be provided for each application or for each protocol used by an application. Thus, server modules may be installed for particular applications (e.g., Microsoft Outlook, Netscape Messenger) or particular protocols such as POP (Post Office Protocol), IMAP (Internet Mail/Messaging Access Protocol), SMTP (Simple Mail Transfer Protocol), LDAP (Lightweight Directory Access Protocol), ICAL (a protocol used for calendar applications), etc. In one alternative embodiment an intelligent server may be installed that is capable of handling multiple protocols or requests from different application client modules.

One purpose of operating a server on each device is to allow a user to operate an application with the same functionality, or almost the same functionality, when the device is disconnected (i.e., disconnected from any other device in the system) as when

connected. More specifically, by having server functions installed on the user's device, the device need not connect to a separate computer system in order to manipulate application data.

In addition, the use of application servers in a fully connected network environment allows data to be synchronized in an efficient manner. In particular, without application servers (e.g., in a client/server system), most data operations may result in some network traffic, even read operations that do not cause any change in the data. In a present embodiment of the invention, however, network traffic is generated only when data is changed. In particular, read operations can be satisfied on local devices instead of having to retrieve the data from another device.

Thus, in system 100 of FIG. 1, in which electronic mail data is synchronized among multiple devices, a user of device 100a may, while disconnected from other devices in system 100, draft new messages, read messages from email data 114a, delete messages, rename them, etc. Email server 112a performs these functions at the request of email client 102a. As already described, the user's changes to the email database will be passed on to another device, and other devices' changes will be received, when device 100a connects to another device.

The replication engine installed on device 100a in state 202 is configured to log or record (e.g., in log file 116a) operations performed by email server 112a that alter email data 114a. A suitable form for log file entries is described in the previous section.

In one embodiment of the invention state 202 and/or state 200 may be performed automatically or semi-automatically when device 100a is connected to server 100c. For example, device 100a may connect to server 100c and identify itself as a new member of a particular group of devices among whom data is to be synchronized. Server 100c may then automatically register and configure the device as necessary (e.g., install replication engine 106a, email server 112a and/or email data 114a). During this registration/configuration procedure, device 100a may learn of the other devices in system 100 and a preferred or default node with which to synchronize. Illustratively, each device is assigned a unique name and may belong to one or more groups in which data for one or more applications are synchronized.

In this embodiment of the invention server 100c is always or nearly always online, so that a synchronizing device can always synchronize with at least one other device or access necessary information. For example, server 100c may be configured to keep track

of multiple groups of synchronizing devices and monitor which ones have members connected at a particular time so that this information may be passed on to individual devices as they come online. Thus, whenever a device is ready to synchronize, it may connect to server 100c to determine what devices are available.

In state 204, other devices that will synchronize data within the system are configured. In particular, devices participating in data synchronization with device 100a may be logically grouped together. Each device is then configured so that the other devices can synchronize with it.

Devices having sufficient resources (e.g., storage capacity) may be configured in a similar manner as described for device 100a in state 202. Devices low in resources may be treated differently. For example, a low-resource device such as device 100d, which lacks sufficient resources to locally store a full copy of the data being synchronized, depends upon another device (e.g., device 100b, server 100c) to access the shared data. A low-resource device may store a subset of the data locally (e.g., the most-used portion of the data). Whether or not a low-resource device stores a local copy of the shared data, in one embodiment of the invention it must store and maintain a log file in order to record data changes it performs. In an alternative embodiment, however, the log file may be limited in size (e.g., not store as many entries as other devices) or the low-resource device may record entries in another device's log file.

In optional state 206 device 100a is disconnected from any other devices in the system that it may have been connected to for configuration, synchronization or other purposes. Device 100a may have been connected to another device, for example, in order to replicate and create a first local version of data for an application. In particular, device 100a may have created email data 114a by copying email data 114c from server 100c or email data 114b from device 100b. A first replication of the data may occur during state 202 or at some other time before device 100a is disconnected. Illustratively, when making a first copy of a set of synchronized data, the replication engine on the sending device simply transmits the full set of data stored on the device at that time. If device 100a was fully configured without being connected to another device then state 206 is unnecessary.

In state 208 a user operates email client 102a on device 100a (and/or any other applications on the device having data that is shared with other devices) while device 100a is disconnected from any other devices in system 100. Due to the presence of email

server 112a, data manipulation requests from email client 102a are serviced locally using email data 114a. This arrangement allows the user to operate the application in virtually the same manner as if the device was connected to server 100c in a client/server relationship. As one significant difference from a client/server mode of operation, however, data access is much faster because the data is now stored locally. Furthermore, application server 112a may be small in size because it only needs to service requests from one device. Therefore, the server module will be easy to manage and will operate efficiently.

In state 210, while the user operates email client 102a, replication engine 106a logs all changes to email data 114a into log file 106a. In this embodiment the email server notifies the replication engine of transactions that must be logged. Illustratively, each entry in a log file relates to one or more transactions or operations in which the data was altered in some manner. Thus, each entry captures sufficient information to allow another device to determine whether the transaction conflicts with any other data changes and to locate where the altered data is stored (if not included in the log entry). A log file entry may include information such as the time of the transaction(s), the device that performed a transaction, the identity of a message that was affected, the action that was performed on the message, etc.

In state 212 device 100a is connected to server 100c, device 100b or another device in system 100. The user may, for example, connect the device to another system employed by the user, such as a desktop computer or workstation.

In state 214, device 100a transmits to the other device the changes it made to email data 114a while disconnected. In particular, the entries it made in log file 116a since the last time it replicated or synchronized data are transmitted so that the devices may synchronize their data. The following section describes one method of data synchronization in detail.

In one embodiment of the invention data synchronization is transitive in nature. In other words, device 100a only transmits to the connected device the data changes that it believes have not been recorded by the connected device. For example, device 100a may synchronize with server 100c after a first period of disconnected operation and may later synchronize with device 100b after a second period of disconnected operation. If, however, server 100c and device 100b synchronize before devicc 100a and device 100b synchronize, server 100c will pass on the first set of data changes made by device 100a.

Then, during the second synchronization device 100a need only transmit to device 100b the data changes that device 100a made during the second period of disconnected operation.

In state 216 device 100a receives data changes from the connected device if the connected device knows of any data changes that occurred since the last time device 100a replicated or synchronized its data.

Then, in state 218 device 100a applies to email data 114a some or all of the data changes it received from the other device. As described in the following section, several criteria may be analyzed to determine which changes are to be implemented, the order in which they are to be implemented, and whether any changes should be ignored. In particular, in one embodiment of the invention the replication engine may examine the timestamp of each log entry it receives from other devices to help determine whether two or more data changes conflict. However, a timestamp is not enough to efficiently and accurately synchronize data among multiple devices. For example, one device in the system may delete an email message some time after one or more other devices rename, move, mark or perform some other operation(s) on the message. In this case the operations other than the deletion may be ignored since, in the end, the message should be removed from all email databases. Therefore, in this embodiment of the invention application-specific information may be considered when applying other devices' data changes to a local copy of the synchronized data. As already described, a device may be configured to recognize a certain priority among data operations (e.g., delete operations outrank modifications). Alternatively, some or all conflicts may be presented to a device's user to let him or her decide how they should be resolved.

One skilled in the art will appreciate that FIG. 2 illustrates just one manner in which an embodiment of the present invention may be implemented or operated. Many other suitable methods may be derived from the illustrated method and accompanying description without exceeding the scope of the invention.

Data Synchronization in One Embodiment of the Invention

This section describes method of synchronizing data among a pair of devices in accordance with an embodiment of the invention. This method may be applied when one of the devices connects to the other after a period of disconnected operation. The method may also be applied on a recurring basis to form a regular pattern of data synchronization.

For example, electronic mail data may be synchronized every ten to twenty minutes, while other types of data (e.g., shared files, calendar data) may be synchronized less frequently (e.g., every thirty minutes).

Illustratively, during a synchronization event in one embodiment of the invention the participating devices exchange log file entries describing data changes that they know of. Each device then accepts or rejects each received log entry and enters in its own log file the acceptable entries. Any subset of the data changes corresponding to the accepted log entries may then be applied to each device's local copy of shared data. Conflict identification and resolution processes may be applied to determine which log file entries and data changes to accept and which to reject.

In one embodiment of the invention each device in a system for synchronizing data stores information indicating how recently devices synchronized or received data changes from another device. In one implementation a table, matrix or two-dimensional array is stored, such as table 300 of FIG. 3. In FIG. 3, the names of the devices participating in a group of synchronizing devices are used as both the horizontal and vertical indices. Illustratively, the vertical index indicates the sending device of a pair of devices while the horizontal index indicates the receiving device.

In FIG. 3, each cell of table 300 is configured to store the timestamp of the latest (i.e., most recent) log file entry created by the sending device that is known to have been provided to the receiving device. Thus, if we assume that time table 300 is DEVICE_1's local time table, then Timestamp23 in FIG. 3 represents the timestamp of the most recent data change performed by DEVICE_2 that DEVICE_3 has been informed of, as far as DEVICE_1 knows. The log entry may have been directly provided by the sender to the receiver or may have made its way to the receiver indirectly (i.e., through one or more other devices). If data synchronization events and time table updates could be performed nearly instantaneously, every device in a system would store identical tables. However, this may be unlikely because one or more devices may operate in a disconnected mode. Each timestamp stored in a given time table thus indicates the oldest log entry that is positively known to have been provided to a device. It is possible that later ones have also been provided but that the device on which the time table is stored has not been made aware of that yet. As described below, at some point in each data synchronization event the participating nodes will merge or update their time tables.

When two devices (e.g., DEVICE_1 and DEVICE_3 of FIG. 3) connect and are about to synchronize their data, each device refers to its own time table to determine what log entries it should send to the other device. Thus, in one embodiment of the invention DEVICE_1 scans the column under DEVICE_3 to determine the oldest (i.e., least recent) log file entry that has been provided to DEVICE_3 by any device. DEVICE_1 then sends all entries in its log file that have timestamps newer (i.e., more recent) than that timestamp. In one alternative embodiment, DEVICE_1 simply checks Timestamp13 to determine the oldest log entry made by DEVICE_1 that has been provided to DEVICE_3 and then sends all log entries newer than this.

In one embodiment of the invention a device may transmit log entries to more than one other device at a time. Illustratively, device 100a of FIG. 1 may send entries to device 100b and device 100c by analyzing its time table and locating the most recent log entry timestamp that it can be certain both devices have received. Device 100a may then broadcast all log entries newer than this one for receipt by device 100b and device 100c. If device 100b or device 100c receives a log entry that it has already recorded or rejected, it simply discards it.

After connecting to another device for synchronization purposes and examining its time table to determine what entries to send, a device transmits the pertinent log file entries to its synchronization partner. The partner reviews the log entries it received, identifies conflicts (if any) between the newly received entries and existing entries in its log file, reconciles any conflicts and then records the new entries that are acceptable.

As already described, log file entries in one embodiment of the invention do not include the actual data changes. Instead, after or while recording newly received entries a device determines (e.g., based on its configuration and/or level of resources) which of the data changes that it has just learned about should be applied to its local version of the shared data. After making this determination the device may connect (or use an existing connection) to a device storing a data change, retrieve the change and apply it locally.

After updating their local data, in the presently described embodiment of the invention the synchronizing devices merge their time tables by comparing values for one or more cells and storing the more recent timestamp of the two.

FIG. 4 is a flowchart depicting one method by which device 100a may connect to device 100b (both shown in FIG. 1) and update its data to reflect data changes learned of from device 100b.

In state 400, an application or utility shared among multiple devices is operated on device 100a. This may occur while device 100a is connected or disconnected from other devices in its group or system.

In state 402, data (e.g., a file, electronic mail message, calendar or address entry, bookmark) used by the shared application is altered in some manner. As described previously, only transactions or operations that change the data must be described to other devices. Each data change is logged to a local log file by a replication engine operating on device 100a.

In state 404, device 100a connects to device 100b (if not already connected) in order to synchronize its data. The connection may already exist if, for example, device 100a is configured to synchronize on a regular basis (e.g., based on time or use of a shared application). Even if device 100a was operating in a disconnected mode in states 400 or 402, it may be configured to automatically connect to another device and synchronize with similar programmable regularity. Alternatively, a user of device 100a may decide to update the data on the device or provide his/her changes to other devices.

In state 406, the replication engine on device 100a consults a local table or other data structure to determine which log entries it should send to device 100b. In one embodiment device 100a simply examines the timestamp identifying the latest log entry it knows that device 100b has received. Device 100a would then provide all log entries after this one to device 100b. In another embodiment of the invention, device 100b scans its time table to find the oldest (i.e., least recent) log entry provided to device 100b by any device. Device 100a then locates that entry in its own log, or locates when that entry would have been stored, in order to provide all subsequent log entries to device 100b.

In state 408, the log entries identified by device 100a are transmitted to device 100b. Each entry may contain just metadata describing an associated data change or may also include some or all of the altered data. In one embodiment of the invention a log file entry contains application-specific information that may help the receiving device identify and/or resolve conflicting data changes.

In state 410, log entries are received from device 100b. The log entries may correspond to data changes made by device 100b and/or changes performed by other devices that are being relayed through device 100b.

In state 412 device 100a identifies any actual or potential conflicts between the data changes it has already recorded (and applied to its local data) and changes described by the newly received log entries.

In one embodiment of the invention a replication engine includes instructions for identifying and/or resolving conflicts. Illustratively, each type of operation (e.g., new, delete, modify, rename) that can be performed on the data of a particular application is identified. The replication engine module then identifies the types of operations (or specific operations) that may constitute a conflict. For example, where an electronic mail application shares an electronic mail database, a replication engine may identify as an actual conflict a situation where it has already recorded and applied a deletion of a mail message (e.g., while the host device was operated in a disconnected mode) and then receives a log entry indicating that another device simply moved the same message (e.g., from an inbox to a particular folder).

As another example, two different devices may both alter or create a particular virtual (i.e., shared) file. Or two devices may independently alter a particular address book entry. One skilled in the art will appreciate that numerous types of conflicts may be identified for any set of shared application data and will understand how they may be identified. Potential conflicts may be numerous, but may be uncovered by examining every possible combination of operations that may be performed on a set of data.

Thus, for each log file transaction received in state 412, the replication engine on device 100a searches its log file for conflicts. Conflicts are then resolved, in any one of a variety of ways. For example, a device may be configured to report all or a subset of all potential and/or actual conflicts to a user and allow the user to choose which, if any, data change to apply. The device may instead be configured to report conflicts involving particular operations (e.g., deletions, edits) to the user. Alternatively, the device may be configured to always apply data changes in the order performed (e.g., by their timestamps) or to resolve just certain types of conflicts by applying the data changes in chronological order. Or, if two sequential or successive operations conflict, the later one may be ignored or un-done in favor of the first.

As another alternative, a device may be configured to favor certain types of operations based on some desired criteria (e.g., a particular application, operation, time, device that performed an operation). A device may, for example, be configured to always apply deletions regardless of the conflicting operation or to favor deletions in certain

instances. A device may be configured to always or selectively favor local operations, operations by other devices, operations by a particular device, etc.

In one particular embodiment of the invention a replication engine is configured to give special consideration to application or operation-specific information. For example, an earlier (time-wise) relocation of an electronic mail message may be discarded in favor of a deletion because a deletion is more final and/or is unlikely to be made without some consideration. Similarly, where a file system (or part of a file system) is being synchronized, if one version of a file is separately modified by two different devices, this will generate a conflict. Instead of simply applying the modifications in their order of occurrence, it may be realized that one or the other user involved in making the changes may not have done so (or may have made different changes) if he/she knew of the other user's changes. In this case the conflict may be presented to a user for resolution.

One skilled in the art will understand that a device may be configured with simple or complex rules for resolving some or all conflicts between data changes performed at different times and/or on different devices. Present embodiments of the invention are intended to allow flexibility in defining, identifying and resolving conflicts. One who creates a replication engine and/or other components of a system described above will readily identify and appreciate the various configuration options that may be adopted.

In state 414 the replication engine of device 100a decides which data changes to apply, based on the received log entries, any conflicts that were identified and/or rules established by a user or system administrator. Thus, the configuration of a device may determine which changes are to be applied. A user of one device may specify, for example, that all changes are to be applied on the device. A device having very little storage may, instead, be configured to avoid implementing any changes if it has to access another device's local version of the shared data. Yet other devices may be configured to apply changes relating to certain applications and/or operations, but not for others. Various criteria may be used to identify which changes should and should not be applied. Such criteria may include the size of a data change, the type of file involved, the device that made the change, the time of the change (e.g., how old it is), the amount of storage space on a device, etc.

In state 416, device 100a retrieves the data changes (e.g., new electronic mail message, altered file, new calendar entry, renamed address entry). Device 100a may need to make connections to devices storing the changes. As already described, log file entries

identify the device performing each data change, thereby informing device 100a where to look for a change. If the device that made the change is unavailable, device 100a may attempt to retrieve a data change from another available device.

Then, in state 418, the data changes are applied to the local version of the shared data stored on device 100a.

Finally, in state 420, device 100a and device 100b update or merge their time tables to reflect the log entries they have exchanged. They may, for example, exchange full or partial tables so that each may replace older timestamp values with newer ones.

Although not depicted in FIG. 4, device 100b will receive log entries from device 100a, identify and resolve conflicts, and apply data changes similar to the manner described for device 100a. One skilled in the art will appreciate that FIG. 4 describes just one method of applying an embodiment of the present invention to synchronize data between two participating devices.

The foregoing descriptions of embodiments of the invention have been presented for purposes of illustration and description only. They are not intended to be exhaustive or to limit the invention to the forms disclosed. Many modifications and variations will be apparent to practitioners skilled in the art. Accordingly, the above disclosure is not intended to limit the invention; the scope of the invention is defined by the appended claims.

What Is Claimed Is:

1. A method of updating shared data on one of multiple computing devices, wherein the shared data is used by applications executing on each of the multiple computing devices, the method comprising:

operating an application server module on the one computing device, while the device is disconnected from the other multiple computing devices, to perform a first transaction on a local version of the shared data in response to a request from an application client module;

describing said first transaction in an entry recorded in a local log file;

connecting the device to another of the multiple computing devices;

determining a last local log file entry known to have been provided to the other computing device;

sending to the other computing device one or more entries recorded in said log file after said last entry;

receiving a remote log file entry from a log file of the other device;

recording said remote entry in said local log file; and

applying to said local version of the shared data a second transaction corresponding to said remote entry.

2. The method of claim 1, further comprising:

identifying a conflict between said second transaction and a third transaction described in an entry in said local log file;

examining application-specific information included in said remote entry; and

determining from said application-specific information whether to apply said second transaction.

3. The method of claim 1, wherein said application server module includes a replication module.

4. The method of claim 1, wherein said operating an application server module comprises:

accepting user commands at an application client module;

receiving service requests at an application server module, from said application client module; and

servicing said requests by accessing and changing a subset of the shared data;

wherein said application client module, said application server module and said subset of the shared data are maintained on the device.

5. The method of claim 1, wherein said describing comprises storing one or more of: a time at which said transaction was recorded, an identifier of the device, an identifier of an application using the shared data, a type of transaction, and application-specific information that may be used to identify or resolve a conflict between said transaction and another transaction.

6. The method of claim 5, wherein said describing further comprises storing a portion of the shared data that was changed during said transaction.

7. The method of claim 1, wherein said determining a last log file entry comprises identifying a timestamp of an entry is said log file pertaining to a previous transaction known to have been provided to the other device.

8. The method of claim 7, wherein said sending comprises selecting one or more log file entries recorded in said log file after said timestamp of said entry pertaining to said previous transaction.

9. The method of claim 1, wherein said applying comprises:

retrieving from the other device a change to the shared data described in said remote entry; and

altering said local version of the shared data in accordance with said change.

10. The method of claim 1, wherein the shared data is used by a first application executing on the one computing device and is used by a second application operating on another of the multiple computing devices.

11. A computer-implemented method of synchronizing data shared among

multiple devices operating one or more applications that use the shared data, comprising:

maintaining a log file on a first device, said log file including a first record corresponding to a first change to the shared data;

receiving at said first device a connection from a second device, wherein said second device operated a first application prior to said connection while disconnected from every other device in the multiple devices;

receiving a second record from said second device corresponding to a second change to the shared data performed on said second device;

examining said second record to determine if said second change conflicts with said first change;

determining whether to apply said first change to a first version of the shared data stored on said first device; and

retrieving said second change from said second device.

12. The method of claim 11, wherein said maintaining a log file comprises operating a replication module to record descriptions of changes to the shared data performed by said first device.

13. The method of claim 11, further comprising:

establishing a connection to a third device; and

transmitting said second record to said third device.

14. The method of claim 13, wherein:

said establishing a connection to a third device comprises establishing connections to a third device and a fourth device; and

said transmitting said second record comprises transmitting said second record to said third device and said fourth device.

15. The method of claim 12, further comprising:

operating an application server module on said first device, with which to manipulate said first version of the shared data in response to a request from an application client module;

wherein said application server informs said replication module of said changes to

the shared data performed by said first device.

16. The method of claim 15, wherein said application server module comprises said replication module.

17. A computer readable storage medium storing instructions that, when executed by a computer, cause the computer to perform a method of synchronizing data shared among multiple devices operating one or more applications that use the shared data, the method comprising:

maintaining a log file on a first device, said log file including a first record corresponding to a first change to the shared data;

receiving at said first device a connection from a second device, wherein said second device operated a first application prior to said connection while disconnected from every other device in the multiple devices;

receiving a second record from said second device corresponding to a second change to the shared data performed on said second device;

examining said second record to determine if said second change conflicts with said first change;

determining whether to apply said first change to a first version of the shared data stored on said first device; and

retrieving said second change from said second device.

18. A computer readable storage medium containing a data structure configured to describe a first change to a set of data shared among multiple computing devices, the data structure comprising:

a timestamp configured to indicate a time at which the first change was recorded in a log file;

a device identifier configured to identify a device on which the first change occurred;

an application identifier configured to identify an application designed to use the shared data;

an operation identifier configured to identify a type of the first change; and

application-specific information that may be useful in resolving a conflict between

the first change and a second change.

19. The computer readable storage medium of claim 18, wherein the data structure further comprises data encompassing the first change.

20. A system for synchronizing data among multiple computing devices, each of the computing devices comprising:

a processor configured to execute an application that uses data shared among the multiple devices;

an application server configured to manipulate a local version of said shared data;

a change log configured to store descriptions of changes made to said local version of said shared data; and

a replication module configured to store entries in said log file pertaining to data changes originated by said application server;

wherein said processor executes said application and said application server manipulates said data while the device is disconnected from the other multiple computing devices; and

said replication module transmits one or more entries in said log file to a second of the multiple devices when the device is connected to the second device.

21. The system of claim 20, wherein the device receives from the second device log file entries corresponding to a set of data changes applied to a version of said shared data stored on said second device; and

said set of data changes include data changes originated by the second device and data changes originated by a third device.

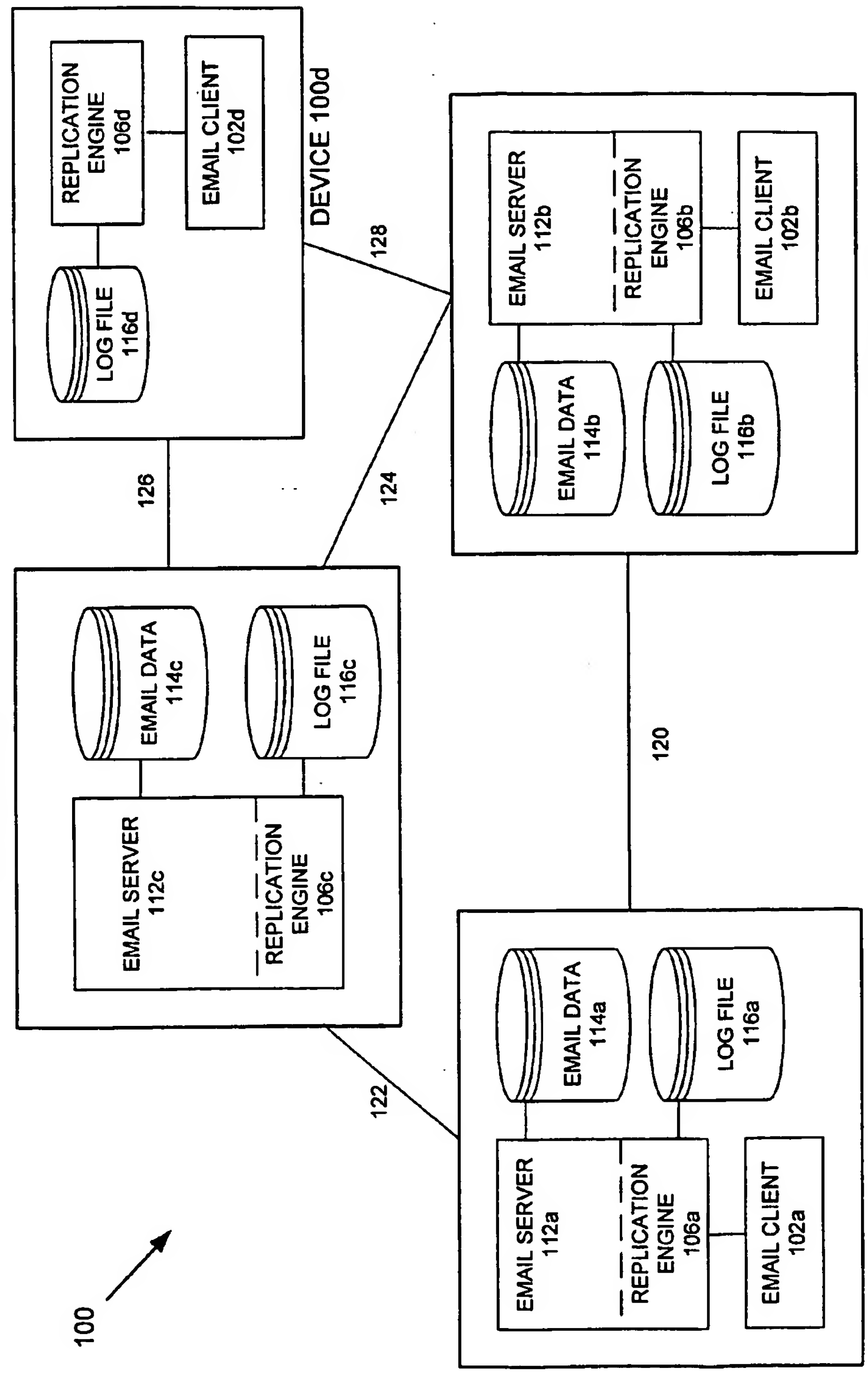

DEVICE 100d
REPLICATION ENGINE 106d
EMAIL CLIENT 102d
LOG FILE 116d
DEVICE 100b
EMAIL SERVER 112b
REPLICATION ENGINE 106b
EMAIL CLIENT 102b
EMAIL DATA 114b
LOG FILE 116b
SERVER 100c
EMAIL DATA 114c
LOG FILE 116c
EMAIL SERVER 112c
REPLICATION ENGINE 106c
DEVICE 100a
EMAIL DATA 114a
LOG FILE 116a
EMAIL SERVER 112a
REPLICATION ENGINE 106a
EMAIL CLIENT 102a
128
126
124
122
120
100

FIG. 1

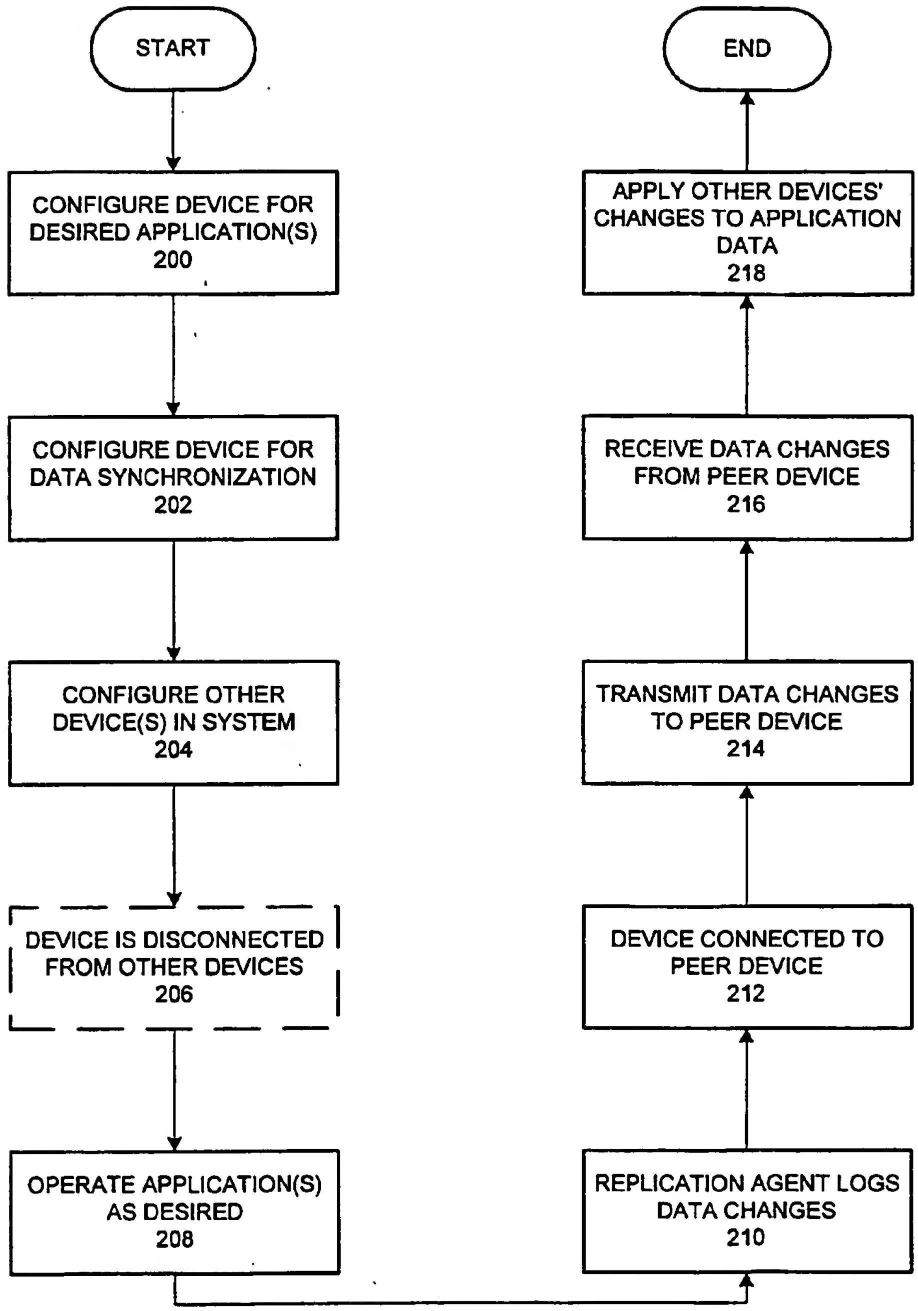

START
CONFIGURE DEVICE FOR DESIRED APPLICATION(S) 200
CONFIGURE DEVICE FOR DATA SYNCHRONIZATION 202
CONFIGURE OTHER DEVICE(S) IN SYSTEM 204
DEVICE IS DISCONNECTED FROM OTHER DEVICES 206
OPERATE APPLICATION(S) AS DESIRED 208
REPLICATION AGENT LOGS DATA CHANGES 210
DEVICE CONNECTED TO PEER DEVICE 212
TRANSMIT DATA CHANGES TO PEER DEVICE 214
RECEIVE DATA CHANGES FROM PEER DEVICE 216
APPLY OTHER DEVICES' CHANGES TO APPLICATION DATA 218
END

FIG. 2

RECEIVING DEVICE

SENDING DEVICE

	DEVICE_1	DEVICE_2	DEVICE_3	DEVICE_4
DEVICE_1	Timestamp11	Timestamp12	Timestamp13	Timestamp14
DEVICE_2	Timestamp21	Timestamp22	Timestamp23	Timestamp24
DEVICE_3	Timestamp31	Timestamp32	Timestamp33	Timestamp34
DEVICE_4	Timestamp41	Timestamp42	Timestamp43	Timestamp44

TIME TABLE 300

FIG. 3